Simulation-based Approaches to Studying Effectiveness of Moving-Target Network Defense

Rui Zhuang[1], Su Zhang[1], Scott A. DeLoach[1], Xinming Ou[1], and Anoop Singhal[2]

[1]Kansas State University
[1]{zrui,zhangs84,sdeloach,xou}@ksu.edu
[2]National Institute of Standards and Technology
[2]psinghal@nist.gov

Abstract

Moving-target defense has been hypothesized as a potential game changer in cyber defense, including that for computer networks. However there has been little work to study how much proactively changing a network's configuration can increase the difficulty for attackers and thus improve the resilience of the system under attack. In this paper we present a basic design schema of a moving-target network defense system. Based on this design schema, we conducted a simulation-based study to investigate the degree to which proactively changing a network's various parameters can decrease an adversary's chance for success. We believe this is an important first step towards understanding why and how the concept of a moving target can be successfully applied to computer network defenses.

1 Introduction

In cyber space, attackers have an asymmetric advantage in that they have time to study our networks to determine potential vulnerabilities and choose the time of attack and gain the maximum benefit. Additionally, once an attacker acquires a privilege, that privilege can be maintained for a long time without being detected [5]. The static nature of current network configuration approaches has made it easy to attack and breach a system and to maintain illegal access privileges for extended periods of time. Thus, a promising new approach to network security has been suggested called the moving target defense (MTD) [1]. While there are many facets of MTD, for computer networks, one can broadly interpret MTD as the fact that the network constantly changes to reduce/shift the attack surface area available for exploitation by attackers. Here, the attack surface consists of the system resources exposed to attackers (e.g. the software residing on the hosts, the ports open to communicate between hosts, and vulnerabilities in the various components) as well as compromised network resources that can be used to further penetrate the system. While promising, there is little research to show that MTDs can work effectively in realistic networked systems. In fact, the approach is so new that there is no standard definition of what an MTD is in the context of network defense, what is meant by attack surface, or metrics to define the effectiveness of such systems.

We believe a set of objective analytical models should exist to predict the effectiveness of MTD systems to protect computer networks. Ideally, these analytical models would be useful at both design time and runtime. As inputs to the models, a set of objective metrics are required that capture specific information related to the exploitable features of the system. The metrics must capture (1) the area that an attacker must search to determine the configuration of the system, (2) the modifiable aspects of the system, and (3) what is changing in the system configuration and how fast the configu-

ration is changing. While the metrics capture what and how the system is adapting, they should be relatable to the effort required by an attacker to attack the system. Based on the configuration of the network being defended, the analytical models should capture the basic steps required to attack the system and determine the effectiveness of a proposed MTD system to defeat attacks attempting to exploit both known as well as unknown vulnerabilities.

In this paper we present our preliminary design of a network moving-target defense system. Based on the design schema, we design analytical models and methods with an eye towards capturing the effectiveness of the proposed MTD system. We then conduct simulation-based experimentation to examine quantitatively the assumption that the MTD system can decrease attacker's success likelihood. We believe such an analytical step is critical before one sets out to build a MTD network defense system. This is just our first step towards understanding and quantifying the impact of moving target defenses on computer networks. Our current analytical model is preliminary and only captures the attacker's perspective. A more comprehensive analysis will also take into account the cost and overhead of deploying the MTD system on the mission the network supports. We will continue our research into a more comprehensive understanding and more realistic estimation of MTD's effectiveness on computer network defense, based on the insights gained from this initial first step.

2 MTD design principles

An MTD system is generally portrayed as a system that adapts randomly over time to make the network configuration appear chaotic to a potential attacker. An architecture of such a system is shown in Figure 1. Here, an Adaptation Engine orders (what appears to be) random adaptations to the network configuration at random intervals. These adaptations are carried out by a Configuration Manager who controls the configuration of the Physical Network. However, adaptations that were truly random in nature could quickly yield the system inoperable since services could be assigned to inappropriate

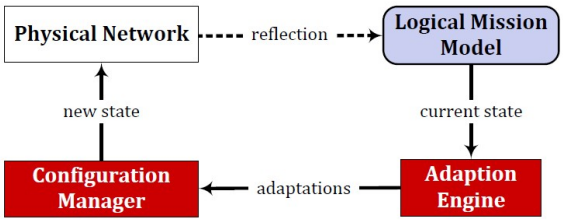

Figure 1: Basic MTD system

hosts or the communications paths required for the system to work appropriately could be interrupted. To enable apparently random adaptations work effectively, the underlying MTD system must have an understanding of the functional and security requirements of the system. In our system, this understanding is based on a Logical Mission Model that reflects the current Physical Network configuration as well as the functional and security requirements of the network. With this information, the MTD system can make apparently random adaptations with an understanding of the requirements of the system and the current configuration.

An alternative vision for MTD systems combines apparently random changes with intelligent control. In this version, adaptations can be selected either randomly or based on risk indicators such as vulnerability scanning results and IDS alerts. The basic concept combines purely random adaptations with fully reactive intrusion response systems, where all responses are made in terms of adaptations. The use of intelligent control techniques allows the MTD to react to suspected intrusions instead of simply adapting randomly. However, addition of the random adaptations allows the MTD to effectively mitigate unpredicted attacks as well as mask the actions of the intelligent control system. By incorporating reactions into adaptations, the system can react to suspected intrusions much sooner than a normal intrusion response system since even responses to false positives will leave the system in an operational state with no more overhead expended than for a random adaptation. The basic architecture of an intelligent MTD system is shown in

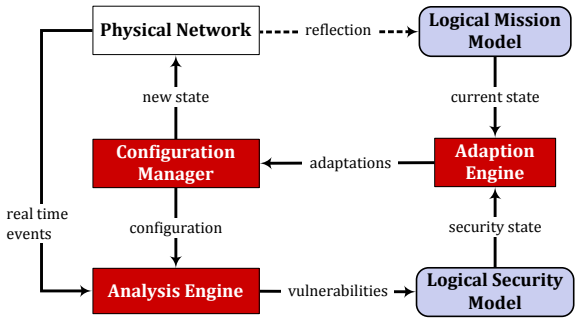

Figure 2: Intelligent MTD system

Figure 2, which extends the basic MTD system of Figure 1. Figure 2 adds two new entities and extends the function of the existing entities. The basic operation of the random adaptation remains the same. However, we have added an Analysis Engine that takes real-time events from the Physical Network and the current configuration from the Configuration Manager to determine possible vulnerabilities and on-going attacks. The Adaptation Engine now looks at the network's current state along with the security state to determine if there is a security issue that needs to be addressed via a system adaptation. If so, a set of adaptations dealing with the security issue is sent to the configuration manager in addition to other random adaptations.

Adapting a network's configuration will impact attack success for two main reasons: 1) the attacker needs to spend more time canvassing the network in order to identify topological information (both physical and logical) that will be useful in further attacks, and 2) the attacker cannot keep privileges gained for long and will have to frequently regain privileges. However, these reasons for success also imply two inherent challenges for an MTD system. First, the MTD system must provide legitimate users and applications with a way of locating required resources in the midst of the adaptations. However, once a user account or application is compromised, the attacker would gain this ability to locate resources, thus limiting the effectiveness of MTD systems. To address this challenge, the MTD system must limit the damage incurred by a compromised system component by limiting the knowledge the component has regarding the adaptation process. That is, a compromised user or application will not give an attacker complete knowledge of the locations of other resources. Nevertheless, an MTD will not be invincible simply because it is moving. An attacker can still incrementally accumulate knowledge and privileges as long as he is not detected. Thus rigorous analysis is needed to understand to what extend MTD can reduce the likelihood an attack can succeed in reaching its goal. The second challenges is that, while the MTD system can adapt by moving or "refreshing" an application or resource (e.g. using a fresh clean virtual machine (VM) to replace an existing VM), the transition process will likely disrupt services and introduce a necessary overhead cost. Thus, an understanding of the effect of adaptations on both the system performance and security improvement must be understood so that appropriate trade-offs can be made.

2.1 Proof-of-concept MTD system

Due to the lack of existing MTD systems to analyze, we must design a parameterized proof-of-concept MTD as a first step towards understanding and validating the technical merits of a MTD network defense system. While existing research on adaptive defense systems have focused on modification of single low-level aspects of a network such as IP addresses, our vision is to develop a framework in which multiple aspects of the system can be modified simultaneously. Aspects that we plan to consider include IP addresses, ports, firewall settings, host-application assignments, application types/versions, and protocols. An overview of our framework is shown in Figure 3, which is similar to Figure 2 with details added. We illustrate our approach with a simple example that supports a mission planning system as well as allowing users to access mission related e-mails. The mission planning system accesses three different databases: an asset database that includes the types and numbers of assets available to carry out mission planning, a target database that includes the latest intelligence on targets of interest, and a geographical database that includes maps and related geographical infor-

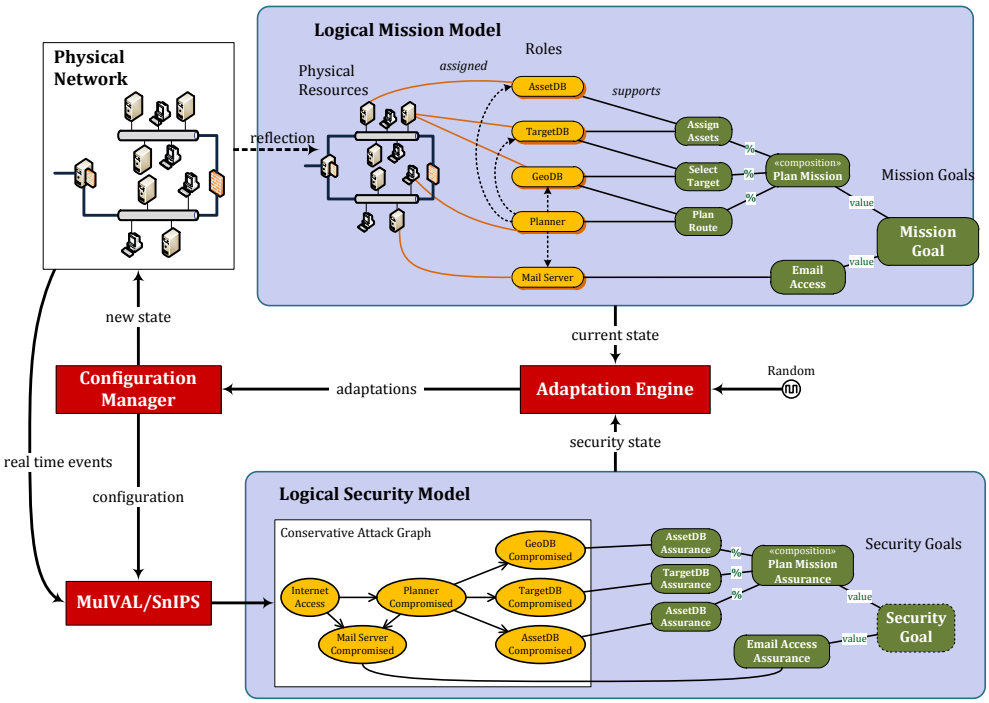

Figure 3: Design scheme for a network MTD system

mation about the areas required for planning appropriate ingress, target attack, and egress routes. We assume the planner also requires access to e-mail to coordinate with other organizations and users.

A key concept of our design is the abstract *Logical Mission Model* that captures the network resources, the services used, and the dependencies between services that are required to achieve the overall mission of the network. The Logical Mission Model is based on DeLoach's Organizational Model for Adaptive Computational Systems (OMACS) [7]. OMACS is a model based on human organizations that allows intelligent reasoning algorithms to assign *agents* to play *roles* in an organization in order to achieve specific organizational *goals*. Agents can only be assigned to a role if they possess all the *capabilities* required by that role. The OMACS model is general-purpose and has been successfully applied to multiagent systems, cooperative robotics, and distributed sensor networks. In this research, services are the network "roles" such as the Planner, AssetDB, TargetDB, and GeoDB. These roles support the main "goals" of the network, which include allowing users to plan missions (Plan Mission) and allowing users to access e-mail (Email Access). In the implementation each role is instantiated on an physical or virtual host, which equate to OMACS "agents". In addition, required network communications between the roles is also specified in the Logical Mission Model.

The moving-target mechanism is created by reassigning physical (or virtual) resources (agents) to various roles as required to support the goals of the mission goal model. This process is carried out by the Adaptation Engine, which can be based on existing OMACS-based reorganization algorithms. These are closely related to traditional algorithms for allocating single-agent tasks to single-task agents, for which efficient suboptimal algorithms exist [7, 8, 20]. The Adaptation Engine determines an acceptable assignment of roles to physical resources based on the security state or a ran-

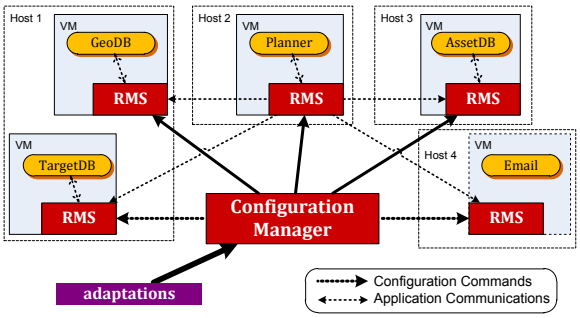

Figure 4: Resource Mapping System

dom trigger, role requirements and computational capabilities required. An *acceptable assignment* is a near optimal assignment that is sufficiently different from the previous set of assignments so as to ensure the adaptation appears chaotic to an attacker.

2.1.1 Resource Mapping System

The reason for adapting can be either (1) purely random or (2) based on identification of possible attacks or known vulnerabilities by the Analysis Engine, which produces a *Conservative Attack Graph* that indicates such potential threats. A conservative attack graph's nodes represent captured assets by attackers. Unlike traditional attack graphs, a conservative attack graph will err on the security side and assume there is an attack path between any two assets as long as the attacker is able to identify the target asset from the source asset. Since the system is constantly adapting, the mapping from a role to the actual resources used to instantiate the role, including its IP addresses, is also constantly changing. Thus in general an attacker will not be able to launch an attack from an asset to any other asset. However, the assets need to perform their expected functionalities and for this reason they need to know the assets with which they need to communicate with (the dotted lines in Figure 4). There needs to be a mechanism to enable such legitimate locating of resources. In our design we call it the Resource Mapping System (RMS).

The purpose of the RMS system is to serve as a security policy enforcement unit for each role. It is best implemented as a hardened system component. As shown in Figure 4, the role of the RMS module in our MTD design is two-folded:

1. It coordinates with the Configuration Manager, which pushes the configuration to various resources. All communication between system services must go through the RMS so that communications can be maintained even as the location of the services change.

2. In an attempt to access and exploit services, the attacker may either (1) follow the RMS or (2) try to guess their locations. The first option forces them to follow a predefined pattern that significantly simplifies intrusion detection and prevention. The second forces them to repeatedly conduct extensive reconnaissance to re-identify service locations, thus increasing the attackers' effort and the likelihood of revealing themselves.

We envision that each critical role will be assigned to a single VM, which will have a dedicated RMS to handle communication with other critical roles. Each RMS will only know the locations of the roles it needs to communicate with as defined by the communication requirements of its associated role. The RMS could be transparent, working as an IP-layer proxy, or provide an API for sending network packets to abstract resources. In either case, all communications between mission-critical roles are controlled by the RMS even as their locations change dynamically.

A drawback to the RMS occurs if attackers compromise a critical role/VM. In this case other roles for which the compromised role initiates communications can be easily located and attacked since the compromised role's RMS knows their location. However, the attacker must follow the exact communication pattern defined by the role model, thus dramatically reducing the potential attack surface. Here, adaptation comes to the rescue. Eventually, the VM of the compromised role will change and the attacker will lose any gained privileges. In this case, even low-confidence alerts could be used to trigger adaptations. We believe that the RMS system is best implemented as a thin layer between the

virtual machine monitor (VMM) and the VM's operating system, so that a compromised VM cannot directly corrupt the integrity of the RMS.

The actual reconfiguration of the physical resources is carried out via a Configuration Manager, which must be highly secure since obtaining it would enable an attacker to determine the system configuration quickly at any time. The Configuration Manager will work closely with the RMS to handle the communication among the critical services.

2.2 Conservative Attack Graph

An integral aspect of MTDs is that an attacker must continually re-gain the knowledge and privileges obtained through prior attacks. For example, VM refreshing will eliminate all privileges gained on the VM effectively forcing the attacker to take a step back in the plan toward the goal. This effect invalidates the typical monotonicity assumption [2] found in most attack-graph works where an attacker cannot lose a privilege after gaining it. In an MTD system, it becomes important to model losing privileges due to constant changes in the system configuration. The frequency of such MTD mechanisms will affect how far an attacker can move forward in a system. Modeling such dynamism requires a state-machine model, rather than the commonly used dependency attack graphs [10, 12, 14]. Previous state-enumeration attack graphs [15, 19] have encountered scalability challenges when applied to large networks [14]. However, for analyzing the MTD effect on computer networks, we do not need to apply a fine-grained attack-graph model. Thus, we propose a *conservative attack graph*, which assumes the existence of unknown vulnerabilities without enumerating every one of them. This assumption actually makes the state model smaller and will likely lend itself to stochastic analysis.

As an example, Figure 5 shows the conservative attack graph for the mission planning system depicted in Figure 3. The topology of the conservative attack graph is partially derived from the role model that supports the mission goals. As shown in Figure 3, the Planner role initiates interactions (depicted by the arrows between roles) with

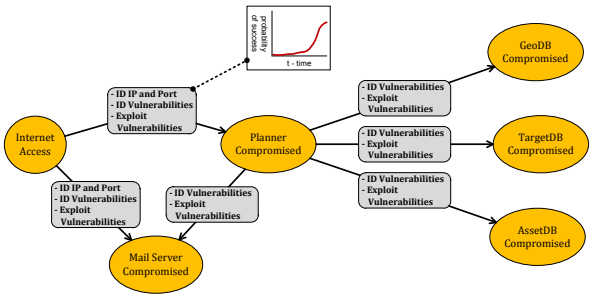

Figure 5: Conservative Attack Graph

the AssetDB, TargetDB, GeoDB, and Mail Server roles. In our MTD system design, this security policy is enforced by the RMS explained above. The only legitimate access paths in the system are (1) from the Internet to the Planner or the Mail Server and (2) from the Planner to the Mail Server and the three database servers (AssetDB, TargetDB, and GeoDB). The conservative attack graph captures these logical access paths.

A key motivation of MTDs is that if an attacker deviates from the presumed access paths, e.g. by guessing wrong the location of the service to attack, it will fall into a decoy that can issue alerts and track the attacker's activities. The RMS components on the virtual machines implement the network communication policy (derived from the role model) to adhere to the logical paths. If an RMS component is compromised, the attacker would be able to bypass this control and try to access a service that is not exposed to the VM. However, in such situations the compromised RMS (and the attacker) would not know the location of those services and thus the attacker would have to correctly guess the IP address and port (among other aspects) of the next target, which is a low-probability event. Thus we can assume that a successful attack must follow the predefined service access paths, which dramatically reduces the attack surface of the system.

The conservative attack graph can be viewed as a state-transition system. Each arrow is annotated with a label describing the activities involved to move from one state to the next. The effort involved in the activities can be measured in various ways.

For example, one can ascribe a success-likelihood to time diagram to indicate how much time it will take the attacker to reach a certain success likelihood for a specific action. From each state, there will also be a probability for the attacker to be forced to "move back" to one of the prior state along the path, due to the MTD mechanisms. These will be added as additional transitions among states when designing metric models.

3 Simulation-based Experiments

To determine if our approach has merit, we devised a high-level simulation that reflects a MTD system. The simulation testbed was built on an existing network security simulator called NeSSi2 [18]. NeSSi2 is an open-source, discrete-event based network security simulator with extensive support for constructing complex application-level scenarios based on a simulated TCP/IP protocol stack. Figure 6 shows our simulated network topology created in NeSSi2.

3.1 Simulation Assumptions

As the first step in our simulation-based research, we made a number of simplifying assumptions in the adversary model and adaptation mechanisms. Such limitations are not inherent in the MTD system design and will be removed in our future research.

3.1.1 Assumptions on adversary model

We assume that once a node is compromised, the attacker can immediately use the RMS to attack the next node in the attack path. The attacker is assumed to know the basic system architecture as defined by the Role Model and thus the attack is restricted to the four VMs assigned to the four roles. The attacker is assumed to know immediately when a resource it has compromised has been refreshed.

3.1.2 Assumptions on the adaptation mechanisms

All adaptations are applied at a specified time interval and are random in nature. Adaptations are limited to VM refreshing, and all VMs assigned to a given role have the same configuration except for its ID and IP address.

While these assumptions make the simulation easier, they are also tilted in favor of the attacker. First, we assume only a simple (as opposed to intelligent) MTD system. Second, we only use VM refreshing thus we do not assume any variability of software versions, operating systems, etc. that would make compromises more difficult. We also assume the attacker knows the system design and that the attacker can immediately compromise the RMS thus allowing the attacker to immediately attack the next target.

3.2 Moving mechanisms and attacks

The three main components of our testbed include the *Defense component*, the *Attack component* and the *Ground Truth* component. The Defense component contains the Configuration Manager, three physical resources (hosts) and four active VMs. These four VMs can be assigned to any host to play any of the four roles: Planner, TargetDB, AssetDB, or GeoDB. The Configuration Manager is the core of the Defense component and combines the functionality of the Configuration Manager and the Adaptation Engine from Figure 3. At each simulation time interval, the Configuration Manager selects an adaptation by creating a *new* task, $t_{new} = \{role, host, vmid, ip\}$, by (1) randomly picking a role, (2) randomly picking a host, (3) generating a new unique VM ID, and (4) randomly picking an unassigned IP address. The Configuration Manager finds the associated old task, $t_{old} = \{role, host', vmid', ip'\}$, within its set of existing tasks, T, by matching role names. It then informs the old task's current host, $host'$, to shut down the $vmid'$ VM and tells the new host, $host$, to start up a new VM at address ip to play the $role$. Finally, the Configuration Manager updates the Ground Truth component with the current configuration.

The *Attack component* is responsible for simulating the attack and makes extensive use of the CAG, which allows the attacker to know exactly where to attack in order to achieve its overall attack goal (to compromise the TargetDB). The Attack component uses the CAG shown in Figure 7 to guide the attack. As we can see, the only available attack path is to penetrate from Internet to Planner,

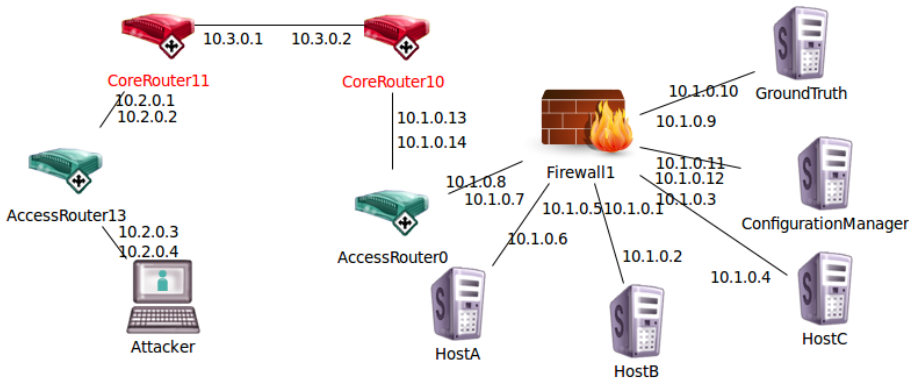

Figure 6: Network Topology

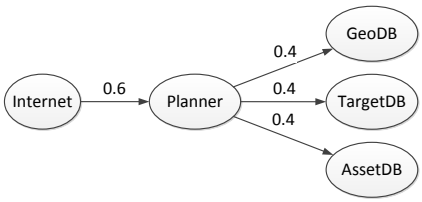

Figure 7: Conservative Attack Graph

then from Planner to TargetDB. The edge values in the CAG denote the attacker's probability of success of an attack launched from the one node to the next, if both nodes remain static. For example the 0.4 between planner and TargetDB means that the attacker has a 40% chance of compromising the TargetDB if (1) it has already compromised the Planner and (2) the Configuration Manager does not adapt either the Planner or the TargetDB during the time step. In the real system, the edge values will be computed based on combining the probability of unknown and known vulnerabilities of the roles in the current configuration.

Each simulated attack has several steps. First the current CAG is retrieved from the Ground Truth component. Next, after waiting Δt time intervals (which simulates the time required to launch an attack), an updated version of the CAG is retrieved and used to determine whether the attack has succeeded or not. To determine attack success, we first generate a random value and check to see if it exceeds the CAG edge value for the current attack. If it does, the simulation determines if the VMs on either the attacker's current node or the attacked node have been refreshed; if either of them has been refreshed, the attack fails. If the attacker's current node was the VM that was refreshed, the attacker is pushed back to its previous node. If neither were refreshed, the attack succeeds.

The *Ground Truth component* maintains the current CAG and provides the connection between the Attack component and Defense component. The Ground Truth component receives task information from Configuration Manager and updates the CAG as required. It also supplies information from the current CAG to the Attack component when requested.

The Attack component, Defense component, and Ground Truth component are implemented as NeSSi2 components along with the three host resources: hostA, hostB, hostC. These six components are loaded onto the corresponding nodes as shown in Figure 6. The hosts do not actually perform their assigned role responsibilities, but merely exist to give the attacker something to attack. The results of our initial experiments are presented in the next section.

4 Results and Discussion

We conducted several experiments to see how the frequency of system adaptation could impact intrusion attempts. We also included a control experiment where no adaptation occurred. In each exper-

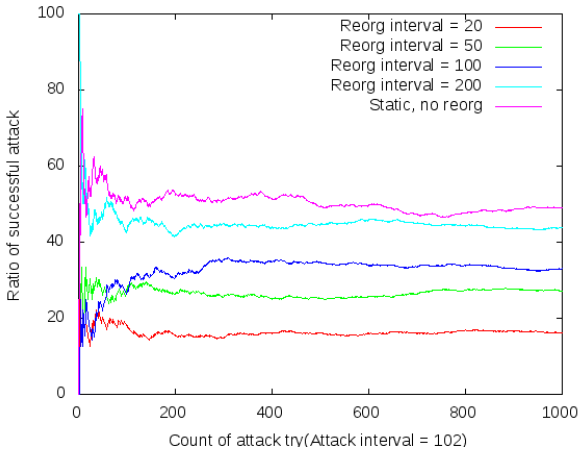

Figure 8: Success of Individual Attacks

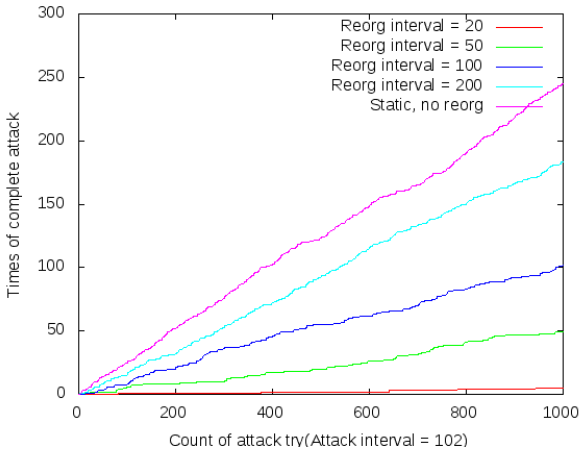

Figure 9: Attacks Completed Against TargetDB

iment, we assume a fixed Δt between each attack of 50 time intervals. However, due to the message delivery delay in the NeSSi2 simulator between the Attack and Ground Truth components (52 time intervals), the *actual* Δt was 102 time intervals. We ran 1000 experiments each for 5 different adaptation intervals (20, 50, 100, 200 and ∞). The adaptation interval corresponds to the time interval between the the Configuration Manager's adaptation (∞ corresponds to the static system).

Figure 8 shows the success of each individual attack between nodes for each adaptation interval. While the lines for each adaptation interval fluctuates initially due to small number of samples, it becomes more stable as the number of attacks increase. We can see that when the configuration remains static, the success ratio of each attack is approximately 50%. However, as the adaptation interval shrinks, the individual attack success ratio also shrinks, eventually reaching 16.2% for a adaptation interval of 20 time intervals.

Figure 9 provides a more complete look at the effect of the MTD as it measures the ability of the MTD to deter a completed attack from the Internet through the Planner to the TargetDB. Figure 9 clearly shows that as the adaptation interval is reduced, the effect of the MTD defense is clearly visible. When the configuration is static, the number of completed attacks (out of 1000) is 245, while an adaptation interval of 100 reduces that number to 50 and an adaptation interval of 20 allows only 5 successful attacks against the TargetDB.

4.1 Discussion

We believe these results demonstrate the promising effectiveness of moving target defense for enterprise computer networks.

The design of our MTD is based on understanding the current situation, which is captured in a set of runtime models. These runtime models allow the system to reason over the current state of the system and produce adaptations to confuse and rebuff potential attackers. The simulation presented here is our first, and one of the first anywhere, simulation of MTD for enterprise network security. As such, the simulation implemented only a simple MTD system and did not demonstrate the full power of an MTD system. However, the results demonstrate a potential effectiveness of MTD for enterprise computer networks and, thus, we plan to continue to make the simulated system more complex, increase the sophisticated of the simulated attacks, and integrate in the full power of an intelligent MTD system.

It is also clear that the simplicity of our example network made the task much easier for the attacker. Since the ability of the attacker to reach a particular node in the network is directly related to the length of the path the attacker must traverse to

reach it, it seems that the longer the path in the CAG to a node of interest, the more protection an MTD system provides. In the future we plan to further investigate this phenomenon to see how it can be used in designing systems that are expected to work in an MTD environment.

5 Related Work

Most of the prior work on MTDs in a network context has been related to low-level techniques such as IP address shifting and network routing and topology control. We discuss several efforts.

Dynamic network address translation. As part of the DARPA Information Assurance Program starting in 1999, BBN developed a dynamic approach to active network defense in order to demonstrate the hypothesis that "Dynamic modification of defensive structures improves system assurance" [11]. Their goal was to inhibit an attacker's ability to map the network, thus making attacks more difficult. Their approach made it appear as though the addresses and port numbers used by the network's computers changed dynamically via Dynamic Network Translation (DYNAT), which disguised host identity information in TCP/IP packets.

BBN and Sandia National Labs ran several experiments of DYNAT's capability to degrade an adversary's ability to map a network. The experiments showed that DYNAT made it almost impossible to map the network while significantly increasing the attacker's effort [11]. Even when teams discovered a host's location or were able to hijack a session, their advantage was time-limited due to dynamic changes to translation seed values. The results also showed that DYNAT highlighted typical attacker actions as obvious anomalies that made spotting the attacks much easier. Beating DYNAT is difficult [16] as a direct assault requires detailed knowledge of the address hopping mechanism as well as the compromise of a trusted device, algorithm, and initial values and keying material. Indirect assaults using phishing schemes could compromise the system and allow an external system to participate in the address hopping. However, there are several drawbacks [13, 16]; DYNAT requires that trusted computers on both sides of the communication be within the protection of DYNAT processes and there are problems related to application interoperability.

Applications that participate in their own defense. In the DARPA's APOD (Applications that Participate in their own Defense) project [4], BBN also proposed port and address hopping techniques to confuse would-be attackers and thus prevent them from identifying and ultimately attacking network computers. Essentially, this approach was similar to DYNAT, except that it applied port and address hopping at layers above TCP such as in CORBA calls and used off-the-shelf utilities for a more general solution. However, the same advantages and problems exist with the APOD approach as with the DYNAT approach.

Network address space randomization. Antonatos et al. use a similar network address space randomization (NASR) scheme to thwart hit list worms [3]. Their implementation, however, is much different as they configure DHCP servers to expire the leases of hosts at various intervals to support address randomization. They consider several different expiration policies including changing host addresses only when hosts are rebooted as well as using timer-based settings. The soft change timer specifies the minimal interval between address changes when there is no activity on the host, while a hard change timer specifies the maximum time a host can maintain a given address, regardless of activity. Several experiments concluded that NASR can be useful for hit list worm defense, although additional research is needed. As with the DYNAT approach above, the researchers found that the approach is also beneficial in making the worms easier to detect.

Dynamic route adaptation. In [6], Compton proposed an approach to dynamically changing network packet routes so that observable traffic patterns change on a regular basis [6]. The goal of this work is to make network mapping more difficult and to make packet sniffing less effective. They developed a new metric for topological network change that captures the difference in the required band-

width between two nodes. This measure was combined with a mixed integer linear programming objective function designed to produce the optimal topology to produce near optimal solutions that are significantly different from the current solution. Results showed that, on average, links were only active 33% of the time. The main drawback to this approach is the time required to come up with solutions, which ranged from minutes to days. To solve this problem, Greve developed a heuristic solution called the Network Obfuscation Heuristic (NOH) [9], which was created by supplementing a greedy pricing algorithm to produce a polynomial-time approximation. Although the experiments to compare the two algorithms are not complete, they do reveal that NOH significantly decreases the runtime in all cases (2 orders of magnitude for networks larger than 20 nodes) and that the deviation from the optimal solution is also acceptable for larger networks.

Proactive obfuscation Proactive obfuscation is an approach to creating almost identical replicas of software applications that share identical functionality with fewer shared vulnerabilities [17]. The approach restarts fresh versions of servers periodically that react differently to identical attacks. Generally, when attacked using known vulnerabilities, the modified servers either do not respond as expected or crash instead of being compromised. The authors propose to create these near identical replicas using semantics-preserving code transformations. The authors showed that with sufficient entropy in the executables, the approach was effective at thwarting known attacks while their approach to automatically generate diverse executables did not greatly increase costs.

6 Conclusion

In this paper we presented a preliminary design of a network moving-target defense system. We conducted simulation-based experiments to study the effects of randomly changing one aspect of the system — role to VM mapping, in reducing attacker's success likelihood. The results show, as expected, reduced attack success likelihood with increasing frequency of changes. This is our very preliminary first step towards building a comprehensive evaluation and analysis framework for network moving-target defense research.

Acknowledgement. This work was supported by the Air Force Office of Scientific Research under award no. FA9550-12-1-0106, and U.S. National Science Foundation under award no. 1038366 and 1018703. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the above agencies.

7 Acknowledgement

This work was supported by the Air Force Office of Scientific Research under award no. FA9550-12-1-0106, and U.S. National Science Foundation under award no. 1038366 and 1018703. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the above agencies.

References

[1] National Cyber Leap Year Summit 2009 co-chairs' report, networking and information technology research and development. Technical report, 2009.

[2] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of 9th ACM Conference on Computer and Communications Security*, nov 2002.

[3] S. Antonatos, P. Akritidis, E.P. Markatos, and K.G. Anagnostakis. Defending against hitlist worms using network address space randomization. *Comput. Netw.*, 51(12):3471–3490, August 2007.

[4] Michael Atighetchi, Partha Pal, Franklin Webber, and Christopher Jones. Adaptive use of network-centric mechanisms in cyber-defense. In *IEEE Intl. Symp. on Object-Oriented Real-Time Distributed Computing*, 2003.

[5] Devlin Barrett. Hackers penetrate nasdaq computers. http://online.wsj.com/article/, February 2011.

[6] Matthew D., Kenneth M. Hopkinson, Gilbert L. Peterson, and James T James T. Moore. Network obfuscation through polymorphic routing and topology control. *IEEE Transactions on Dependable and Secure Computing*, 2012. In preparation.

[7] Scott A. DeLoach, Walamitien Oyenan, and Eric Matson. A capabilities-based model for adaptive organizations. *Autonomous Agents and Multi-Agent Systems*, 16:13–56, 2008. 10.1007/s10458-007-9019-4.

[8] Brian P. Gerkey and Maja J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[9] Gabriel H. Greve. Network Security Toolkit Including Heuristic Solutions For Trust System Placement and Network Obfuscation. Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 2010.

[10] Sushil Jajodia, Steven Noel, and Brian O'Berry. Topological analysis of network attack vulnerability. *Managing Cyber Threats: Issues, Approaches and Challanges*, 2003.

[11] D.L. Kewley and J.F. Bouchard. Darpa information assurance program dynamic defense experiment summary,. *Systems, Man and Cybernetics, Part A: Systems and Humans.*, 31:331–336, 2001.

[12] R.P. Lippmann, K.W. Ingols, C. Scott, K. Piwowarski, K.J. Kratkiewicz, M. Artz, and R.K.Cunningham. Evaluating and strengthening enterprise network security using attack graphs. Technical report, MIT Lincoln Laboratory, 2005.

[13] J. Michalski, C. Price, E. Stanton, E. L. Chua, K. Seah, W. Y. Heng, and T. C. Pheng. Final report for the network security mechanisms utilizing network address translation ldrd project. Technical Report Technical Report SAND2002-3613, Sandia National Laboratories, November 2002.

[14] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *13th ACM Conference on Computer and Communications Security*, Oct 2006.

[15] Cynthia Phillips and Laura Painton Swiler. graph-based system for network-vulnerability analysis. In *NSPW 98: Proceedings of the 1998 workshop on New security paradig*, 1998.

[16] Keith A. Repik. Defeating Adversary Network Intelligence efforts with Active Cyber Defense Techniques. Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 2008.

[17] Tom Roeder and Fred B. Schneider. Proactive obfuscation. *ACM Trans. Comput. Syst.*, 28(2):4:1–4:54, July 2010.

[18] S. Schmidt, R. Bye, J. Chinnow, K. Bsufka, A. Camtepe, and S. Albayrak. Application-level simulation for network security. *SIMULATION*, 86:311–330, 2010.

[19] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, 2002.

[20] Christopher Zhong and Scott A. DeLoach. Runtime models for automatic reorganization of multi-robot systems. In *6th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS 2011).*, May 2011.